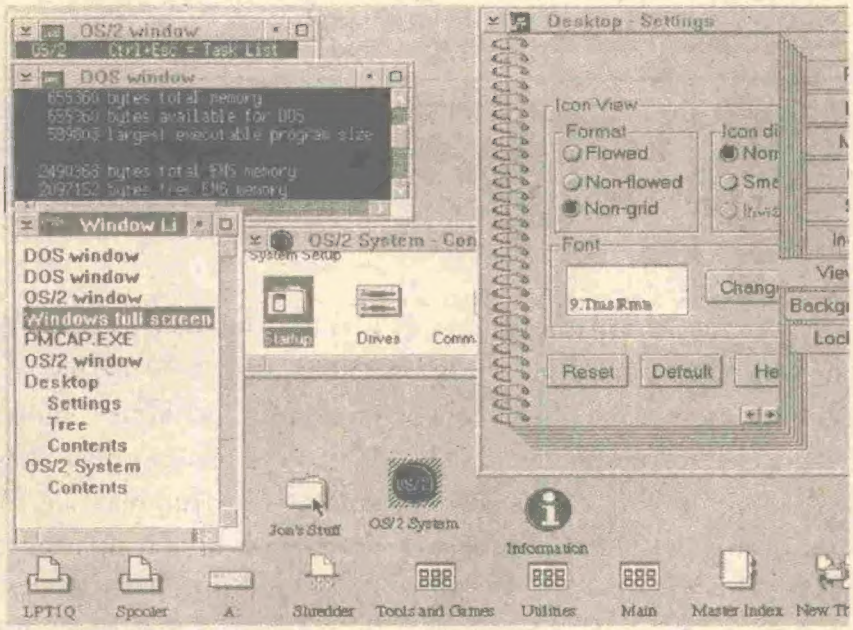
สหัส พรหมสิทธิ์

OS/2 กับเน็ตเวอร์ก(1)

ในการเชื่อมต่อคอมพิวเตอร์เข้าด้วยกันให้เป็นเน็ตเวอร์ก มีอยู่มากมายหลายแบบ ลองใช้ OS/2 ดูแล้วบ้างว่ามีอะไรมากมาย



การรัน OS/2 ในสภาพแวดล้อมที่มีแลน (LAN) เป็นโครงสร้างขั้นพื้นฐานนั้นจะมีเรื่องใหญ่ ๆ อยู่สามเรื่องให้ดำเนินการคือ (1) การนำ OS/2 ไปใส่ไว้ในพีซีของผู้ใช้รายต่าง ๆ (2) ทำความเข้าใจเรื่องของดิสตริบิวเต็ดแอพพลิเคชัน (distributed applications) ที่ใช้ OS/2 และ (3) รู้วิธีทำงานของ OS/2 ในรูปของเซอร์ฟเวอร์แพลตฟอร์ม (server platform) ขั้นแรกก็จะต้องมีการขยายฐานของ OS/2 เพื่อให้สามารถให้การสนับสนุนในแง่ของเน็ตเวอร์กแก่ผู้ใช้พีซีที่อาศัย OS/2 โดยที่แอพพลิเคชันต่าง ๆ ที่รันอยู่ในพีซีจะต้องสามารถเข้าไปใช้ไฟล์ พรินเตอร์ตลอดจนดีไวซ์ต่าง ๆ ที่อยู่ในเน็ตเวอร์กซึ่งจะมีเซอร์ฟเวอร์ร่วมอยู่ด้วย ขั้นต่อไปก็จะต้องให้การสนับสนุน OS/2 แก่ผู้ใช้ทั่วทั้งเน็ตเวอร์ก ตลอดจนการมีฟีเจอร์สำหรับการบริหารระบบที่เป็นพีซี สุดท้ายก็คือ การหาระบบเน็ตเวอร์กที่มีขีดความสามารถในการสื่อสารสูงเพื่อให้ผู้ใช้พีซีบนฐาน OS/2 สามารถเข้าถึงเน็ตเวอร์ก ตลอดจนสภาพแวดล้อมคอมพิวเตอร์อื่น ๆ ที่อยู่คนละแห่งได้

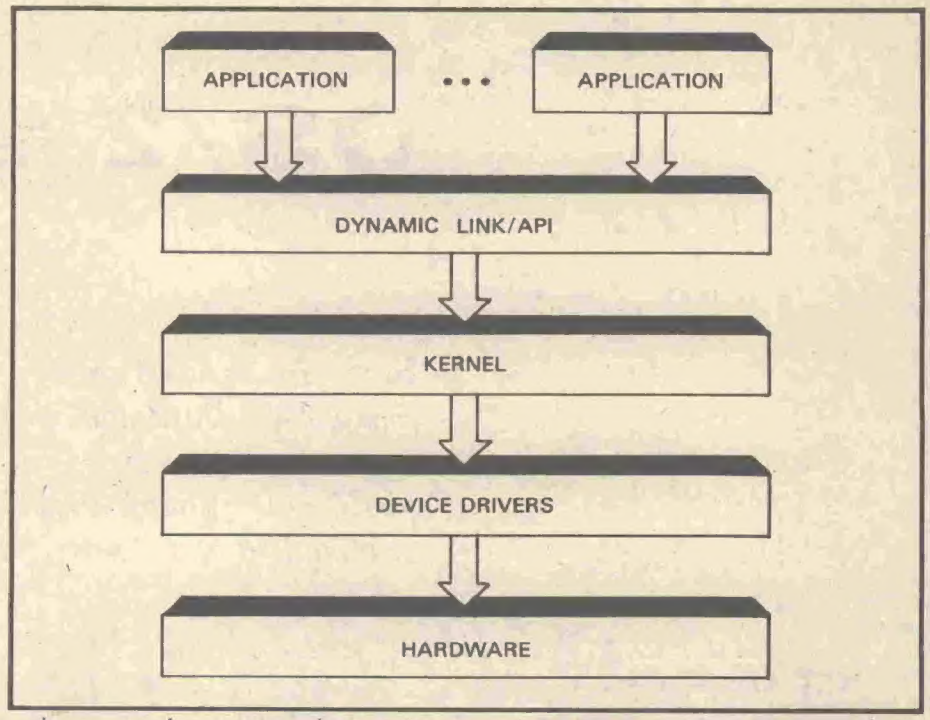
กระจายภาระ

ความสามารถของ OS/2 ในการทำงานแบบหลายภาระพร้อม ๆ กันที่เป็นมัลติทาสกิง (multitasking) กับการทำ OS/2 พีซีให้เป็นเน็ตเวอร์กนั้น

ได้เป็นที่มาของการพัฒนาดิสตริบิวเต็ดแอพพลิเคชันต่าง ๆ อยู่หลายอย่าง ที่มีสถาปัตยกรรมแบบ ยูสเซอร์ (หรือผู้ใช้) กับเซอร์ฟเวอร์ คือในส่วนของผู้ใช้ก็จะมุ่งไปที่เรื่องของ ยูสเซอร์-อินเตอร์เฟส กับการนำเสนอ (presentation) ส่วนในแง่ของเซอร์ฟเวอร์ก็จะมุ่งไปที่การจัดการกับทรัพยากรที่ต้องแบ่งกันใช้ (shared resources) ซึ่งการที่จะทำให้เกิดมีดิสตริบิวเต็ด-แอพพลิเคชันได้ก็ต้องอาศัยองค์ประกอบขั้นพื้นฐานสี่อย่างด้วยกัน คือ

* ระบบดำเนินงานบนพีซีของผู้ใช้
* ระบบดำเนินงานบนเซอร์ฟเวอร์
* เน็ตเวอร์กกิง-โปรโตคอล ซอฟต์แวร์ที่จะทำให้พีซีและเซอร์ฟเวอร์ติดต่อสื่อสารกันได้ และ
* กลไกที่ทำหน้าที่สื่อสารสำหรับอินเตอร์โปรเซส กับอินเตอร์ทาสก์ (interprocess/intertask communications หรือ IPC) ซึ่งในการทำงานนั้น IPC จะอยู่เหนือซอฟต์แวร์ทั้งสามโดยจะวางอยู่บนสุดและจะคอยให้บริการในแง่ของการเรียกขอและการตอบรับระหว่างผู้ใช้กับเซอร์ฟเวอร์

นอกจากองค์ประกอบทั้งสี่ในข้างต้นแล้ว ดิสตริบิวเต็ด แอพพลิเคชันแพลตฟอร์ม ยังต้องมีระบบการกำหนดชื่อแบบดิสตริบิวเต็ด (distributed naming system) ที่ใช้สำหรับเชื่อมโยงผู้ใช้กับเซอร์ฟเวอร์ โดยผู้ใช้มองไม่เห็น นอกจากนี้ระบบชื่ออันนี้ยังสามารถทำให้เกิดมีภาพรวมระหว่างผู้ใช้กับผู้บริหารระบบในด้านดิสตริบิวเต็ด-แอพพลิเคชัน และทรัพยากรเน็ตเวอร์กแพลตฟอร์มเช่นนี้ยังมีหน้าที่อื่นอีกในแง่ของการก่อให้เกิดสภาพแวดล้อมที่จะทำให้ผู้รับผิดชอบทางนี้สามารถบริหารจัดการส่วนที่เป็นแอพพลิเคชันของเซอร์ฟเวอร์จากจุดไหนก็ได้ในเน็ตเวอร์ก ซึ่งก่อนที่จะพูดอะไรต่อไปนั้นก็จะขอเริ่มในจุดที่จะก่อให้เกิดความกระจ่างที่สุดเสียก่อนนั่นก็คือ เรื่องของความสัมพันธ์ระหว่าง OS/2 กับเน็ตเวอร์ก โดยจะพูดถึงสถาปัตยกรรมของ OS/2 เป็นลำดับแรกเพราะจะนำไปสู่การเข้าใจในหลาย ๆ แง่ก่อนที่จะไปถึงเรื่องเน็ตเวอร์ก



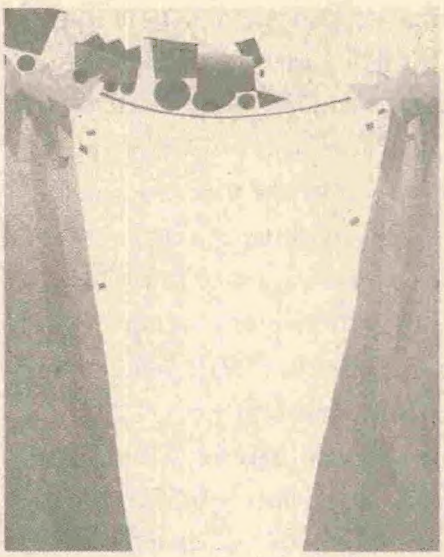
รูปที่ 1 โครงสร้างทั่วไปของ OS/2 ที่มีการแบ่งชั้น (layer) และมีความเป็นโมดูล่าร์มากกว่าดอส (DOS) ที่ทำให้สามารถขยาย, เพิ่มกำลัง หรือสับเปลี่ยนส่วนต่าง ๆ อันนับตั้งแต่ดีไวซ์-ไดรเวอร์กับระบบวิดีโอ รวมทั้งระบบไฟล์ได้อย่างสะดวกโดยไม่ต้องเปลี่ยนเคอร์เนล (Kernel)

OS/2 สถาปัตยกรรมสำหรับอนาคต

งานสร้าง Operating System/2 หรือ OS/2 นั้นน่าจะเป็นการสร้างสรรค์ที่ยิ่งใหญ่มาก ๆ อันหนึ่งของไมโครซอฟต์ (Microsoft) ที่มีจุดมุ่งหมายเพื่อที่จะให้ผู้ใช้แต่ละรายมีเสรีภาพในการใช้คอมพิวเตอร์ได้อย่างไม่อั้น โดยไม่ต้องมีการแบ่งสรรปันส่วนให้ผู้อื่น อันเป็นการใช้งานในรูปของมัลติทาสกิง (multitasking) ที่อยู่ในยุคที่ฮาร์ดแวร์กำลังจะมีราคาลดลงเรื่อย ๆ ในขณะเดียวกันกับที่ เวลาของผู้ใช้และเวลาของโปรแกรมเมอร์มีแต่จะแพงขึ้นทุกวัน ปรัชญาของ OS/2 จึงคำนึงถึงผู้ใช้ที่เป็น single user มากกว่าราคาของฮาร์ดแวร์ ด้วยการนำงานที่มีความเร่งด่วนมาก ๆ ให้มารันในฟอร์กราวนด์ (foreground) โดยให้ความสำคัญของ ทรูพุต (throughput) ให้อยู่ในอันดับรองลงไปส่วนวิธีการสร้าง OS/2 นั้นก็แตกต่างจากดอส (DOS) อย่างมากดังแสดงไว้ในรูปที่ 1 ซึ่งจะประกอบด้วยขั้นของแอพพลิเคชันที่อยู่บนสุดและชั้นล่างสุดก็จะเป็นฮาร์ด-

แวร์ ส่วนตรงกลางจะประกอบด้วยซิสเต็มส์-เลเยอร์ (system layers) ต่าง ๆ อยู่ 3 เลเยอร์คือ (1) ซับซิสเต็มส์ (subsystems) (2) เคอร์เนล (kernel) และ (3) ดีไวซ์ไดรเวอร์ (device drivers) การทำงานของซิสเต็มส์-เลเยอร์ต่าง ๆ ทั้งสามขั้นนี้ แม้ว่าจะเกี่ยวข้องกัน แต่ก็มีเป้าหมายที่แตกต่างกัน ประการแรกก็คือ ซับซิสเต็มส์ในข้อ (1) นั้นจะมีหน้าที่ให้สิ่งที่เรียกว่า แอพพลิเคชัน-โปรแกรม-อินเตอร์เฟส (applications program interfaces หรือ API) ส่วนเคอร์เนลก็จะเป็นผู้ให้บริการที่เป็นขั้นพื้นฐานของระบบ สุดท้ายก็คือดีไวซ์ไดรเวอร์ที่มีหน้าที่ดูแลจัดการกับสิ่งต่าง ๆ ในฮาร์ดแวร์ อันแสดงถึงการออกแบบระบบดำเนินงานที่มาในแนวอันแปลกใหม่ในแง่ที่ว่าแต่ละส่วนของระบบจะทำหน้าที่ที่ตนเชี่ยวชาญและจะส่งข้อมูลต่อไปยังส่วนอื่น ๆ ต่อไป อันเป็นผลของการออกแบบให้เป็นโมดูล่าร์ ซึ่งทำให้การสับเปลี่ยนซับซิสเต็มส์ต่าง ๆ ตลอดจนดีไวซ์ไดรเวอร์ทั้งหลายเป็นไปโดยง่าย เช่นถ้าจะมีการเปลี่ยนเวอร์ชันอันใหม่ก็ทำได้ด้วยการเอาฮาร์ดแวร์ดีไวซ์ชิ้นใหม่มาประกอบเข้ากับของเดิมเสร็จแล้วก็ทำการคัดลอกไดรเวอร์ไฟล์ เข้าไปในเครื่องพร้อมด้วยการเปลี่ยน CONFIG.SYS ไฟล์กับรีบูต (reboot) ระบบใหม่อีกครั้ง ก็จะได้เวอร์ชันใหม่ที่พร้อมที่จะรันได้ทันที ซึ่งจะตรงข้ามกับมินิคอมพิวเตอร์ที่ต้องการการทำ system generation เพื่อที่จะติดตั้งดีไวซ์ตัวใหม่หรืออัพเดตส่วนใดส่วนหนึ่งของระบบดำเนินงาน

ในกรณีของ OS/2 นี้เคอร์เนลจะทำหน้าที่เป็นระบบประสาทส่วนกลางของระบบดำเนินงานในด้าน เมมโมรี-แมนเนจเมนต์ จัดลำดับของงานชิ้น

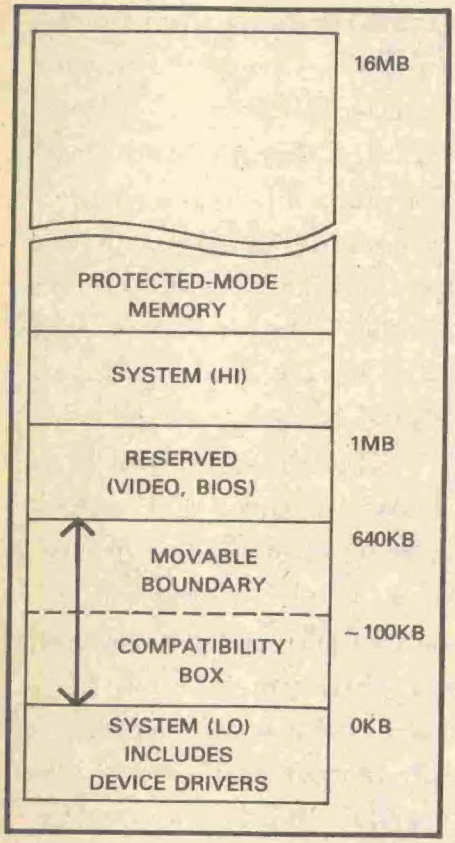


ต่าง ๆ อินเตอร์โปรเซสคอมมิวนิเคชัน ไฟล์ I/O ไทม์เมอร์ เซอร์วิส (timer services) บริหารสภาพแวดล้อมรวมทั้งการอำนวยความสะดวกในด้าน basic text message ฯลฯ จะขอสรุปในขั้นนี้ว่าการออกแบบ OS/2 ให้เป็นโมดูล่าร์และขยายออกไปได้นี้ ได้ก่อให้เกิดการเปลี่ยนแปลงทั้งในด้านของผู้ใช้ และแอพพลิเคชัน ซิสเต็มส์ อินเตอร์เฟส อันทำให้สามารถเพิ่มสิ่งที่เรียกว่ากราฟิก ยูสเซอร์ อินเตอร์เฟส (เช่น Presentation Manager ที่ใช้ร่วมกับ Microsoft Windows) เข้าไปใน OS/2 ได้ รวมทั้งการเพิ่มบทบาทให้ API เพื่อเป็นการเตรียมการไว้สำหรับการให้บริการในด้านการสื่อสารและฐานข้อมูลสำหรับ Extended Edition ของ IBM

ฮาร์ดแวร์กับเมมโมรีแมนเนจเมนต์ของ OS/2

OS/2 ได้รับการออกแบบให้รันกับซีพียู (CPU) แบบ 80286 และ 80386 ซึ่งเป็นสถาปัตยกรรมของอินเทล (Intel) โดยเฉพาะในด้านของสถาปัตยกรรมของระบบดำเนินงานกับขีดความสามารถต่าง ๆ จะผูกพันอย่างล้ำลึกกับ 80286 โดยได้เผื่อช่องว่างไว้บ้าง สำหรับการก้าวไปสนับสนุน 80386 80486 ฯลฯ อันสืบเนื่องจากว่า วิวัฒนาการการออกแบบซีพียูนั้นจะมีต่อเนื่องกันตลอดโดยไม่หยุดยั้ง การเผื่อไว้สำหรับการเปลี่ยนแปลงเช่นนี้จึงน่าจะต้องมีอยู่ในใจของผู้ออกแบบระบบดำเนินงานไม่มากก็น้อย ดังเช่นกรณีของ OS/2

ซีพียูแบบ 80286 และ 80386 สามารถทำงานได้ทั้งในรีลโหมด (real mode) และโปรเทคเต็ดโหมด (protected mode) ในกรณีของรีลโหมดนั้นซีพียูดังกล่าวจะทำหน้าที่เหมือนชิพแบบ 8086 ที่ทำงานได้เร็วมาก ๆ โดยมีเมมโมรี-แอดเดรสแต่หนึ่งเมกะไบต์และจะอาศัยฟิสิกคอล-แอดเดรสเป็นเครื่องกำหนดแอดเดรสในเมมโมรี (หรือหน่วยความจำหลัก) โปรแกรมต่าง ๆ (เช่น แอพพลิเคชันโปรแกรม) หรือส่วนสำคัญ ๆ ของระบบดำเนินงานสามารถที่จะเข้าไปในส่วนใด ๆ ของที่ว่างในเมมโมรีนี้ได้อย่างเสรี ส่วนโปรเทคเต็ดโหมดนั้นจะให้แอดเดรสสเปซ (address space) ได้มากขึ้นคือ แบบ 80386 สามารถใช้รีลเมมโมรีได้ 16 เมกะไบต์ในรูปของแรม ที่โยงต่อกับเครื่องและสามารถใช้เวอร์ชวล (virtual) เมมโมรีได้ 1000 เมกะไบต์ โดยจะได้แอดเดรสสเปซจากการสแวป (swapping) เซกเมนต์ต่าง ๆ เข้าและออกจากดิสก์ แบบ 80386 จะใช้รีลเมมโมรีเป็นจำนวน 4,000 เมกะไบต์และให้เวอร์ชวลเมมโมรีเป็นจำนวน 64 ล้านเมกะไบต์ ซึ่งทั้งหมดนี้ก็เป็นเพียงการมองดูในแง่ของตัวเลข เพราะพีซีส่วนใหญ่มักจะมากับ แรมขนาด 1 หรือ 2 เมกะไบต์ และจะมีเนื้อที่ในฮาร์ดดิสก์ก็ราว 30 หรือ 40



รูปที่ 2 แสดงการใช้งานสำหรับหน่วยความจำหลัก (เมมโมรี) ของซีพียูแบบ 80286 โดยยอมให้มีการนำดอส โปรแกรมต่าง ๆ มาวิ่งใน compatibility box ของ OS/2 ได้ แต่จะมีฟรี-เมมโมรีน้อยกว่าของดอส (DOS)

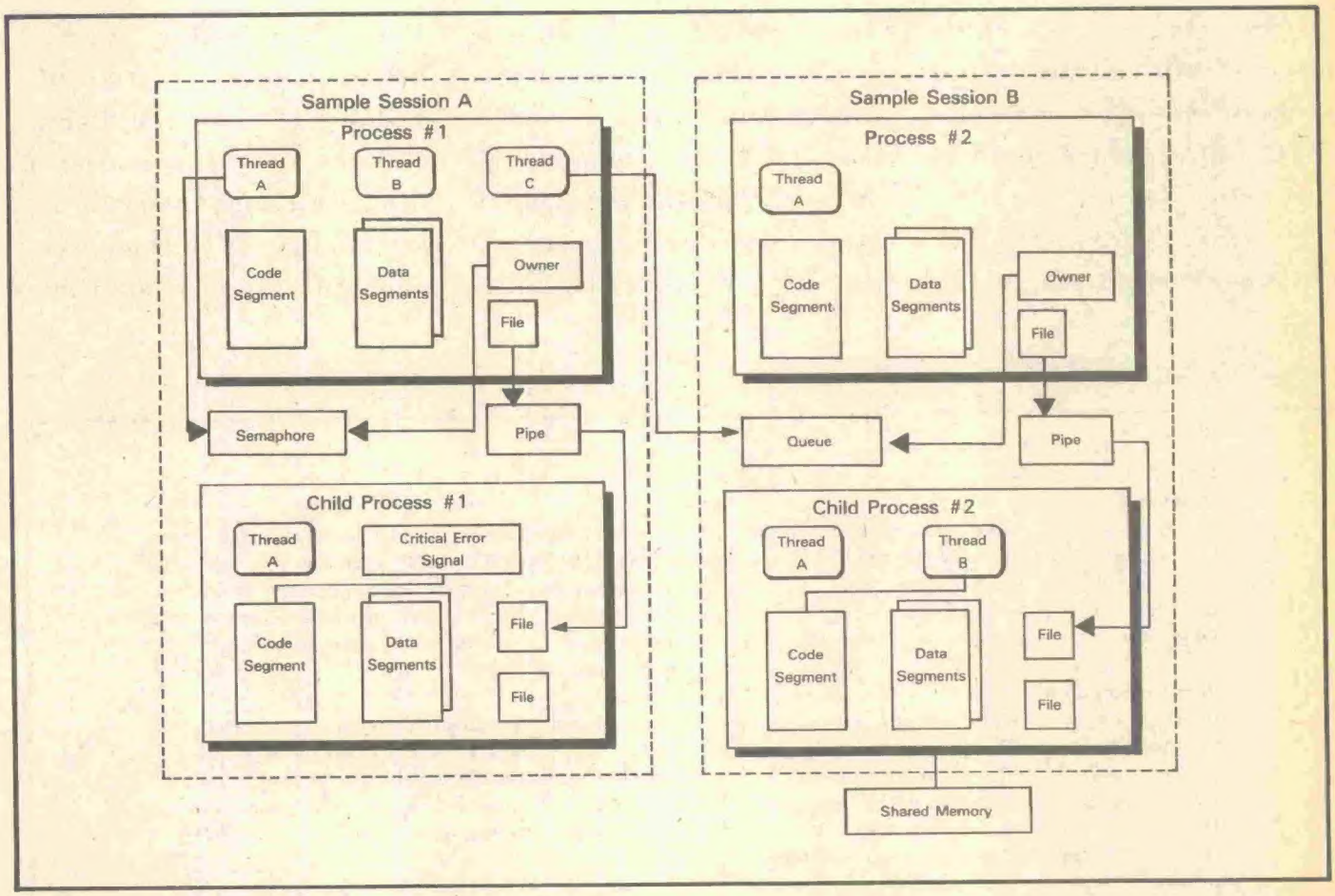
เมกะไบต์ อันเป็นตัวเลขที่แล้ว ๆ มา ซึ่งในขณะนี้ก็จะเปลี่ยนไปอีก ข้อดีของโปรเทคเต็ดโหมดก็คือ การป้องกันไม่ให้มีการขีดเขียนซ้ำซ้อนเกิดขึ้นในเมมโมรี อันเนื่องมาจากการทำงานของโปรแกรมจรจัดที่ไม่รู้ว่ามาจากไหน ซึ่งถือว่าเป็นการป้องกันที่จำเป็นมากสำหรับสภาพแวดล้อมที่เป็นมัลติทาสก์ ทั้งหมดนี้ก็เป็นซีพียูฮาร์ดแวร์อย่างย่นย่อ ซึ่งต่อไปก็จะพูดถึงเมมโมรีแมนเนจเมนต์เฉพาะเท่าที่จำเป็นก่อนที่จะก้าวไปพูดหลักเกณฑ์ของมัลติทาสกิงของ OS/2 ในต่อไป

สิ่งที่ดึงดูดความสนใจผู้ใช้และผู้พัฒนาอย่างมากสำหรับ OS/2 เห็นจะเป็นความสามารถพังกำแพง 640 กิโลไบต์ได้สำเร็จ เพราะแอดเดรสสเปซของ OS/2 สำหรับซีพียูแบบ 80286 นั้นเป็นการทอดเงาจากฟิสิกคอล-แรม (Physical RAM) ขนาด 16 เมกะไบต์ให้ไปอยู่ในรูปของ เวอร์ชวลแอดเดรสสเปซขนาด 16 เมกะไบต์ให้ไปอยู่ในรูปของเวอร์ชวลแอดเดรสสเปซขนาด 1000 เมกะไบต์ ซึ่งในระยะแรกนี้จะยังไม่มีการสนับสนุนจาก OS/2 สำหรับแอดเดรสสเปซที่มีขนาดใหญ่กว่าที่มีอยู่ใน 80386 แผนผังที่แสดงการใช้งานในหน่วยความจำหลักสำหรับ OS/2 จะอยู่ในรูปที่ 2 โดยได้มีการแบ่งส่วนที่เป็นเรสซิเดนต์ (resident) ออกเป็นสองภาคคือ ซิสเต็มส์ (Lo) กับซิสเต็มส์ (Hi) โดยรีลโหมด พาร์ติชันที่ขยายไปถึง 640 กิโลไบต์และสามารถลดขนาดหรือตัดทิ้งไปได้นี้จะขึ้นกับการกำหนดที่มีอยู่ใน CONFIG.SYS ไฟล์เป็นเกณฑ์ อนึ่งการกินเนื้อที่มากถึง 100 กิโลไบต์ สำหรับซิสเต็มส์ (Lo) นั้นได้ทำให้ผู้ใช้ส่วนมากที่อยู่ในรีลโหมดปาร์ติชันเกิดมีความต้องการเนื้อที่ในกรณีนี้ให้มากที่สุดเท่าที่จะทำได้ อันทำให้ต้องใช้แรม (RAM) ไม่ต่ำกว่า 1.5 เมกะไบต์ หรืออาจต้องเพิ่มเป็นสองเท่าสำหรับงานที่เป็นมัลติทาสกิงที่มีประสิทธิผลสูง ทั้งนี้ก็เนื่องจากว่า OS/2 ไม่ได้รับการออกแบบขึ้นมาเพื่อให้ประหยัดฮาร์ดแวร์ ดังจะเห็นได้จากในการที่ทีมพัฒนาของไมโครซอฟต์ได้ใช้ซีพียูแบบ 80386 พร้อมด้วยแรมอีกเป็นจำนวนมากและฮาร์ดดิสก์ขนาดใหญ่ที่ทำงานเร็วมาก ๆ ในการสร้าง OS/2 ขึ้นมา ฯลฯ เมื่อผู้อ่านได้มีความรู้ขั้นต้นเกี่ยวกับฮาร์ดแวร์และเมมโมรีแมนเนจเมนต์อย่างคร่าว ๆ พอที่จะนึกภาพออกว่าอะไรเป็นอะไร ก็จะมาถึงจุดที่เข้าสู่เรื่องของมัลติทาสกิง โดยจะเน้นไปในจุดที่ว่าในสาระสำคัญ ๆ นั้น โมเดลหรือรูปแบบในการทำงานของมัลติทาสกิงเป็นเช่นใดก่อนก้าวไปถึงเรื่องของเน็ตเวอร์กกับ OS/2

มัลติทาสกิงโมเดลของ OS/2

สภาพแวดล้อมที่เป็น OS/2 โดยทั่ว ๆ ไปได้แสดงในรูปที่ 3 ซึ่งเป็นโปรเซสโมเดลที่ประกอบด้วย ระดับในการทำงาน (abstraction levels) ในรูปต่าง ๆ คือ เซสชัน (sessions) โปรเซส (processes) และเทร็ด (threads) กล่าวอีกนัยหนึ่งก็คือว่าได้มีการแบ่งมัลติทาสกิงออกเป็นสามประเภทอันประกอบไปด้วย เซสชัน โปรเซส และเทร็ด ที่มีการควบคุมแบบจากสูงมาต่ำ โดยมีเซสชันอยู่สูงสุดและมีหน้าที่กำหนดว่าจะให้อะไรมาปรากฏบนสกรีน รวมทั้งดูว่าจะส่งอินพุตจากคีย์บอร์ดไปไหน ส่วนโปรเซสก็จะเกี่ยวกับโปรแกรมต่าง ๆ รวมทั้งการจัดสรรทรัพยากรที่เป็นหน่วยความจำที่โปรแกรมเป็นผู้กำหนด สำหรับในขณะเอ็กซีคิวต์ (execution) แต่ละโปรเซสอาจมีจำนวนเทร็ดมากกว่าหนึ่งตัวที่จะทำหน้าที่แยกจากกันเป็นเอกเทศ กล่าวอีกนัยหนึ่งก็คือ เทร็ดก็คือ หน่วยงานเล็ก ๆ หน่วยหนึ่งในมัลติทาสกิง

สรุปก็คือว่า แต่ละเซสชันใน OS/2 จะมีจำนวนโปรเซสตั้งแต่หนึ่งโปรเซสขึ้นไป และในแต่ละโปรเซสก็จะมีเทร็ดที่อยู่ในนั้นและกำลังทำงานอยู่อย่างน้อยหนึ่งเทร็ด ซึ่งแต่ละเทร็ดก็จะสามารถเข้าไปใช้ทรัพยากรที่มีอยู่ในโปรเซสที่ตนประจำอยู่ได้ เมื่อเป็นเช่นนี้ทรัพยากรทั้งหมดสำหรับเทร็ดแต่ละตัวก็จะเป็น



รูปที่ 3 สภาพแวดล้อมของ OS/2

ส่วนหนึ่งของโปรเซส นอกจากนี้ก็จะมีสิ่งที่เรียกว่าไปป์ (pipes) ที่ทำหน้าที่เป็นไฟล์พิเศษเหมือน sequential files ที่สามารถแบ่งการใช้งานร่วมกับ ชายด์ โปรเซส (child process) ได้ รวมทั้งการมีเนม ไปป์ที่จะก่อให้เกิดมีไปป์วางอยู่ในระหว่างโปรเซสคู่ใดคู่หนึ่ง ซึ่งอันนี้ก็เป็นเรื่องของการแลกเปลี่ยนข้อมูลระหว่างโปรเซสโดยอาศัยสิ่งที่เรียกว่าไปป์ (pipes) คิว (queues) และ แชร์เมมโมรี (shared memory) โดยได้แสดงให้เห็นอย่างชัดเจนในรูปที่ 3 ไปป์นั้นเหมาะสำหรับการแลกเปลี่ยนข้อมูลที่เรียงกันเป็นแถวแบบอนุกรม ส่วนคิวนั้นก็จะมีการทำงานที่แตกต่างออกไปอีก คือจะต้องการฟังก์ชันพิเศษชุดหนึ่งซึ่งไม่สัมพันธ์กับไฟล์ต่าง ๆ โดยจะสามารถตั้งชื่อให้คิวเพื่อให้สามารถแบ่งกันใช้คิวในระหว่างโปรแกรมต่าง ๆ ได้ ส่วนข้อมูลของคิวก็จะมีลักษณะเป็นเหมือนข้อความ สุดท้ายก็จะเป็นเรื่องของแชร์-เมมโมรี (shared memory seqments) อันเป็นกลไกชนิดหนึ่งที่มีอยู่ใน OS/2 แต่จะต่างกับของดอส (DOS) ในแง่ที่ว่าโปรเซสอันหนึ่งจะเข้าไปในเมมโมรีของอีกโปรเซสหนึ่งไม่ได้หากไม่ได้รับอนุญาต ซึ่งสิ่งที่จะเข้ามาทำหน้าที่ให้การอนุญาตก็คือ ตัวแชร์เมมโมรีนี่เอง ทั้งหมดนี้ก็เป็นข้ออธิบายสั้น ๆ ว่ากรรมวิธีที่เป็นอินเตอร์โปรเซส คอมมิวนิเคชัน ขั้นพื้นฐานใน OS/2 มีอะไรบ้าง

สิ่งที่จำเป็นต้องพูดถึงอีกเรื่องก็คือ ไดนามิก-ลิงค์ (dynamic-links) ที่เป็นแฟกซิลิตี้อันหนึ่งที่มีอยู่ใน OS/2 และในวินโดว์ (windows) แฟกซิลิตี้นี้จะทำให้โปรแกรมสามารถเข้าถึงซัพพอร์ตฟังก์ชันต่าง ๆ (support functions) ในแง่ของการยอมให้โปรแกรมเรียกซัพพอร์ต รูทีน ได้โดยตรง อันทำให้สามารถลดหรือขจัดโอเวอร์เฮด (over-head) ของฟังก์ชันโค้ดและฟังก์ชัน

ซีเล็กชัน (selection) โค้ดได้ ลิงเกอร์ (linker) สำหรับโปรแกรมของ OS/2 จะเป็นผู้ใส่ชื่อของฟังก์ชันเข้าไปไว้ใน .EXE ไฟล์พร้อมด้วยสิ่งที่จะเอาไว้สำหรับอ้างอิง ฯลฯ

สรุปได้ว่า OS/2 นั้นสามารถจัดเป็น มัลติทาสกิง-โปรเซส โมเดลที่มีอานุภาพสูงอันหนึ่ง พร้อมกับการมีแฟกซิลิตี้และฟังก์ชันต่าง ๆ ที่ทำหน้าที่ในด้านอินเตอร์ทาสก์ (intertask) คอมมิวนิเคชัน ซึ่งโดยทั่ว ๆ ไปก็พอจะถือได้ว่าเป็นระบบดำเนินงานที่ออกแบบขึ้นมาโดยคำนึงถึงความยืดหยุ่น และการสามารถที่จะขยายตัวออกไปได้เป็นสำคัญ ที่น่าสนใจก็คือว่า ชนิดของโปรแกรมที่จะเขียนภายใต้ OS/2 จะเป็นไปในรูปที่ดีกว่าการเขียนโปรแกรมภายใต้ดอส (DOS) ส่วนผู้ที่สนใจอยากจะเห็นรายละเอียดในเรื่องฟังก์ชันต่าง ๆ ของ OS/2 เท่าที่ได้พูดในตอนต้นก็จะดูได้จากรูปที่ 4 และที่ได้พูดมาทั้งหมด

รูปที่ 4 แสดงฟังก์ชันต่าง ๆ ของ OS/2

List of OS/2 Functions

Process Functions		
Dos CWait	Wait for child task termination	Process-related functions can be executed by any thread within the process. The DosExit function can selectively terminate a thread or the entire process and its children. The exit list is a set of functions to be executed when a process is set to terminate. These functions should be short cleanup routines.
DosExecPgm	Execute program	
DosExit	Exit program	
DosKillProcess	Kill process	
DosExitList	Modify termination list	
Thread Functions		
DosEnterCritSec	Enter critical section	Thread-specific functions allow control of a thread's own execution or of other threads within a process. One thread can suspend and resume another, but it cannot terminate it, which is unfortunate. Priority can be set at a process-and thread-level.
DosCreateThread	Create new thread	
DosGetPrty	Get process priority	
DosResumeThread	Continue thread execution	
DosSetPrty	Set process priority	
DosSuspendThread	Suspend thread execution	
DosExitCritSec	Exit critical section	
Session Functions		
DosSelectSession	Select foreground session	Session functions are simple and direct. They do not have to be used, but they allow a process to manage sessions. Sessions allow control of multiple processes, in the same sense that multiple threads are controlled as a single process. However, a session is a more nebulous entity, since the programs within it may not be related.
DosSetSession	Set session status	
DosStartSession	Start session	
DosStopSession	Stop session	
Signal Functions		
DosFlagProcess	Set signal	Signals are process resources that are related to DOS support interrupts such as critical error and control-break. Like DOS interrupts, signals can be turned on and off. Also, signals can only be processed when they are turned on. The DosFlagProcess function can be used on any process whose handle is available to a thread.
DosHoldSignal	Disable/enable signal	
DosSetSigHandler	Set signal handler	
Timer Functions		
DosSleep	Delay thread	General timer operations are available to any thread. Timers can be periodic or singular. Timers can be used to record a duration or cause an event to occur. Timers can be set by one thread and read by another thread within the same process.
DosTimerAsync	Start asynchronous timer	
DosTimerStart	Start periodic timer	
DosTimerStop	Stop timer	
Pipe Functions		
DosMakePipe	Create pipe	This single function is all there is for pipe support, since all other operations are standard file operations. DosMakePipe returns two file handles: one input file and one output file.

รูปที่ 4 (ต่อ)

Queue Functions		
DosCloseQueue	Close queue	Queues are single-reader, multiple-writer interprocess-communication devices. The creator of a queue is able to read the queue. This includes multiple threads within the same process but not multiple processes.
DosCreateQueue	Create queue	
DosOpenQueue	Open queue	
DosPeekQueue	Peek queue	
DosQueryQueue	Query size of queue	
DosReadQueue	Read from queue	
DosWriteQueue	Write to queue	
Semaphore Functions		
DosCloseSem	Close system semaphore	Semaphores are simple entities that can be set and reset. They can be made available to threads outside the process that creates them. Semaphores can be used for mutual exclusion of resources. Semaphores cannot pass any information between the thread that sets it and a thread that clears it. As such, other mechanisms must be used to exchange information.
DosCreateSem	Create system semaphore	
DosMuxSemWait	Wait for one semaphore of a set to be cleared	
DosOpenSem	Open system semaphore	
DosSemClear	Clear semaphore	
DosSemRequest	Request semaphore	
DosSemSet	Set semaphore owned	
DosSemSetWait	Set semaphore and wait for next clear	
DosSemWait	Wait for semaphore to be cleared	

นี้ก็เป็นเพียงต้องการให้ผู้อ่านเห็นภาพกว้าง ๆ ของ OS/2 ว่าเป็นอย่างไร อันเป็นเสมือนการแนะนำก่อนก้าวไปพูดถึงความสัมพันธ์ของ OS/2 ว่าเป็นเช่นใด อันเป็นเสมือนการแนะนำก่อนก้าวไปพูดถึงความสัมพันธ์ของ OS/2 กับเน็ตเวอร์ก ที่จะทำให้ OS/2 เป็นมัลติยูสเซอร์ (multiuser) และมัลติทาสก์ (multitask) พร้อม ๆ กันไป

บทบาทของแลน (LAN) ใน OS/2

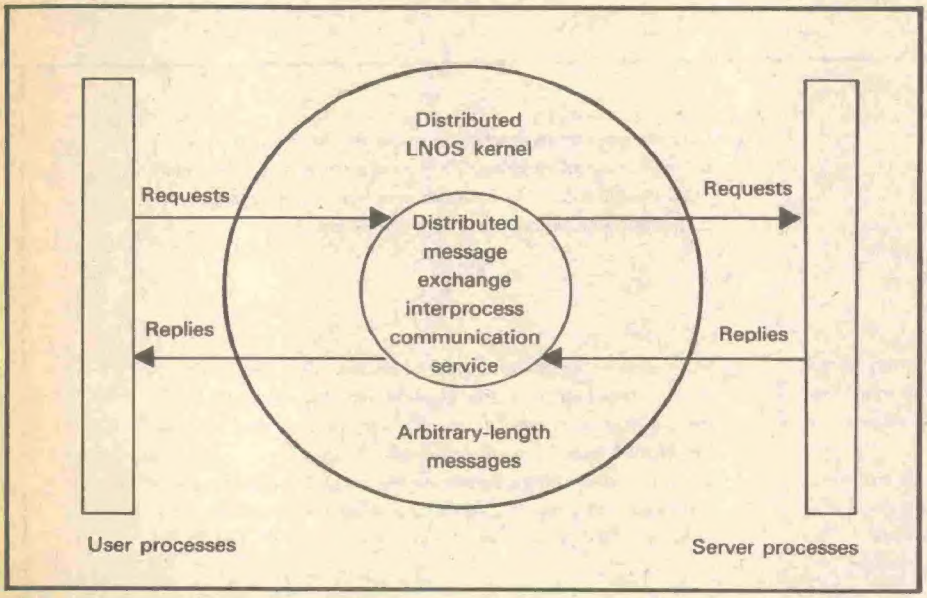
เมื่อใดก็ตามที่ได้มีการพูดถึงเรื่องของเน็ตเวอร์กในบริเวณที่ไม่กว้างมาก นักก็มักต้องพูดถึงแลน โดยเฉพาะในแง่ที่เกี่ยวกับ PC local-area network ซึ่งจะต้องมีซอฟต์แวร์ที่เรียกว่า network operating system (NOS) มาคอยให้การสนับสนุนการทำงานของเน็ตเวอร์กนี้ จุดมุ่งหมายของ NOS ก็คือการก่อให้เกิดความสามารถในด้านมัลติยูสเซอร์ (multiuser) ในหมู่ของเวอร์กสเตชันที่เป็นสถานีงานสำหรับผู้ใช้เดี่ยว กล่าวอีกนัยหนึ่งก็คือ ระบบดำเนินงานสำหรับแลนก็คือสิ่งที่จะเข้ามาแก้ปัญหาเพื่อก่อให้เกิดความสะดวกแบบหลากผู้ใช้ (มัลติยูสเซอร์) โดยยังไม่ได้พูดถึงเรื่องของแอพพลิเคชันแต่อย่างใด หน้าที่ของระบบดำเนินการสำหรับแลนจึงเป็นเสมือนการเป็นภาษารวม ตลอดจนการเข้ามาทำให้มีการเข้าสู่สภาพปกติ เวลามีข้อผิดพลาดหรือเกิดปัญหาในเน็ตเวอร์ก รวมทั้งการก่อให้เกิดความสม่ำเสมอในการปฏิบัติงาน ทั้งในส่วนที่เป็นเฉพาะบริเวณ (local) และส่วนที่อยู่ไกลออกไป นอกจากนี้ระบบดำเนินการนี้ยังเข้ามาช่วยให้สามารถนำแอพพลิเคชันในแลนอันหนึ่งไปรันในแลนอื่น ๆ ได้อย่างไม่มีปัญหาอันไม่ต้องไปผูกพันกับชนิดของฮาร์ดแวร์ จึงสรุปได้ว่าระบบดำเนินงานสำหรับแลนนั้นมีหน้าที่ ดังต่อไปนี้

* แปลงทรัพยากรในด้านฮาร์ดแวร์และซอฟต์แวร์ (ซึ่งจะมีทั้งแลนและพีซี) ให้ไปเป็นทรัพยากรกลุ่มหนึ่งที่เป็นนามธรรม

* ให้การสนับสนุนในด้านการเรียกชื่อ (naming), การแอ็กเซส (Access), แชริ่ง (Sharing), การป้องกัน (Security), การ synchronize การสื่อสารระหว่างกันและการทำให้คืนสู่สภาพปกติเวลาเกิดข้อผิดพลาดหรือเกิดปัญหา

* รวมและจำแนกเพื่อจัดสรรทรัพยากรต่าง ๆ สำหรับพีซีที่อยู่ในเน็ตเวอร์ก

อย่างไรก็ตามระบบดำเนินการสำหรับแลนก็ยังไม่ได้ปราศจากปัญหาเสียทีเดียว เพราะยังไม่สามารถแก้ปัญหาในด้าน data translation กับ data representation ในแง่ที่ว่าส่วนใหญ่ของระบบดำเนินงานเช่นนี้จะให้การสนับสนุนได้แค่เพียงพีซีแบบเดียว และ



รูปที่ 5 โมเดลของระบบดำเนินงานสำหรับแลน

ระบบดำเนินงาน (OS) ที่ใช้ในพีซีเพียงชนิดเดียว นอกจากนี้ยังเป็นการยากที่จะกระจายทรัพยากร (เช่น ไฟล์อันหนึ่ง) ให้ออกไปสู่เน็ตเวอร์กของการให้บริการที่กระจัดกระจาย รวมทั้งโครงสร้างของทรัพยากรที่กระจัดกระจายตลอดจนถึงการทำให้คืนสู่สภาพปกติเวลาเกิดข้อผิดพลาดในระบบดำเนินงานก็พลอยต้องยุ่งยากไปด้วยที่สำคัญก็คือว่า ถ้าระบบดำเนินงานสำหรับแลนไม่มีประสิทธิภาพก็จะกลายเป็นปัญหาคอขวดให้กับระบบอันเป็นปัญหาส่วนรวมทันที ซึ่งเมื่อพูดมาถึงขั้นนี้ก็จะขอนำแม่แบบของระบบดำเนินงานสำหรับแลนมาแสดงให้เห็นในรูปที่ 5 อันเป็นการแสดงลอจิกคอลโมเดล (ligical model) ของระบบดำเนินงานของแลนในแง่ที่ว่าภายใน พีซี-แลน ส่วนใหญ่จะมีคอมมิวนิเคชัน-เคอร์เนล (communication kernel) กระจายอยู่ในพีซีทุกตัว โดยไม่ว่าพีซีนั้นจะเป็นเซอร์ฟเวอร์ (server) เวอร์กสเตชันหรือเป็นทั้งเวอร์กสเตชันและเซอร์ฟเวอร์

จุดมุ่งหมายของโมเดลที่แสดงไว้ในรูปที่ 5 นั้นจะเป็นไปในรูปของกลไกร่วม เพื่อแสดงให้เห็นถึงการสื่อสารระหว่างผู้ใช้แลนว่ามีวิธีติดต่อสื่อความกับเซอร์ฟเวอร์ได้อย่างไร คือในขั้นแรกนั้นผู้ใช้ก็จะส่งความต้องการออกไปจากพีซีหรือเวอร์กสเตชันซึ่งก็จะผ่านเข้าไปในคอมมิวนิเคชัน เคอร์เนลที่อยู่ในพีซีของผู้ใช้แล้วไปสู่เน็ตเวอร์ก จากนั้นก็จะเข้าไปในเคอร์เนลที่อยู่ในเซอร์ฟเวอร์ที่จะทำหน้าที่อย่างใดอย่างหนึ่งให้ ต่อมาผู้ใช้ก็จะได้รับคำตอบจากเซอร์ฟเวอร์ซึ่งจะเดินทางโดยผ่านกลไกแบบเดียวกันนี้ อันแสดงว่าอย่างน้อยจะต้องมีซอฟต์แวร์ชิ้นเล็ก ๆ หนึ่งชิ้น (เช่น เคอร์เนล) ที่คอมพิวเตอร์ทุกตัวในแลนต้องอิงอยู่ นอกจากนี้ก็ยังมีกลไกร่วมอื่นอีก เช่น อินเตอร์เฟสของผู้ใช้ โดยอินเตอร์เฟสของผู้ใช้ในระบบดำเนินงานสำหรับพีซี-แลนที่ถือว่าเป็นขั้นดีนั้น จะต้องมีคุณสมบัติขั้นต่ำสุดที่ประกอบด้วยหัวข้อต่าง ๆ ดังนี้

* เปิดโอกาสให้สามารถเลือกว่าจะใช้ระบบแบบเมนู (menu driven) หรือระบบแบบคอมมานด์โหมด (command mode)

* ตัวเซอร์ฟเวอร์จะต้องสามารถโหลด ดีฟอลต์ (default) ซอฟต์แวร์ให้ตัวของมันเองในทุกครั้งที่ผู้ใช้ล็อก (log) เข้าไปในเน็ตเวอร์ก

*สามารถเอ็กซีคิวต์ เซอร์ฟเวอร์คอมมานด์ ภายในเทอร์นคีย์ (turn key) แอพพลิเคชัน เพื่อไม่ให้ผู้ใช้ต้องยุ่งกับรายละเอียดของเน็ตเวอร์ก

* มีสิ่งช่วยผู้ใช้ในรูปของออนไลน์ (on-line help) ทั้งในด้านแอพพลิเคชันและในด้านเน็ตเวอร์ก ตลอดจนการมีวิธีที่จะกลับคืนสู่สภาพปกติเวลาเกิดข้อผิดพลาดในแลนที่ต้องเป็นไปอย่างมีประสิทธิภาพสูง

* มี locator service ที่สามารถบอกผู้ใช้ว่า ทรัพยากรอันไหน (อย่างเช่น ไฟล์อันใดอันหนึ่ง) อยู่ตรงไหนในแลน รวมทั้งสามารถสนับสนุนการใช้งานพีซีแบบที่ไม่มีดิสก์

สรุปแล้วถ้าสามารถเลือกใช้อินเตอร์เฟสที่มีคุณสมบัติตามที่ได้กล่าวไว้ในข้างต้นได้ ก็จะเกิดความคล่องตัวในการสื่อความรวมทั้งการทำงานในแลน ต่อไปก็จะวกเข้าสู่เรื่องของ OS/2 กับเน็ตเวอร์ก ชนิดสมบูรณ์แบบ เนื่องจากได้มีการกล่าวถึงสิ่งต่าง ๆ ที่เป็นพื้นฐานไว้พอแล้ว